GRAPHENE: GRaph-based Analysis and Prediction Harnessing Efficient Natural language Exploration in Data Science

Minseo Yoon[1] **Jiwon Jeong**[1] **Seonga Choi**[1] **Eunjin Kim**[1]
2021320322 **2021320303** **2021320301** **2021320307**
[1] Department of Data Science
{cooki0615, jjwon4086, 0925star, std50218}@korea.ac.kr

Abstract

In the rapidly evolving domain of data science — a blend of computer science, statistics, and mathematics — information overload has become a significant barrier to learning and comprehension. Faced with this issue, Large Language Models (LLMs) can offer assistance with their capacity to store, utilize, and reason with information; however, LLMs present limitations in attaining expertise across specific domains. In this project, we introduce the **GRAPHENE** methodology to address the challenge of effectively comprehending and managing domain knowledge. GRAPHENE enhances LLMs for data science Question Answering (QA) tasks through advanced fine-tuning, innovative prompting techniques, and structured reasoning with knowledge graphs and graph neural networks. Our results demonstrate marked improvements over existing baseline (Llama2 7B), achieving a 0.22 higher score on open QA tasks, and an increase in accuracy by over 20% on the MMLU dataset. This work shows GRAPHENE's potential to significantly advance LLMs in handling complex, domain-specific queries.

1 Introduction

In the dynamic field of data science, a mixture of computer science, statistics, and mathematics, researchers and practitioners are surrounded by the sweep of data that presents a formidable challenge in its management and comprehensibility. While the advent of search engines has greatly expanded access to the knowledge, acquiring expertise in the nuanced domains of data science remains a complex task that extends beyond simple retrieval of information.

In this study, we suggest that specially trained language models, or large language models, could be a game-changer for data science studies. These LLMs can do more than just look for information; they can analyze and find patterns in the mountains of data we deal with every day. By learning from various data science materials, such models could help us by creating in-depth reviews, guides, and summaries all by themselves. They might even help us see where things are going in the field by looking at both simple and complex data.

Returning to the foundational objectives of computational design — to help handle and make sense of too much information — we introduce a tool designed for data science. This tool aims to boost human knowledge, make team research easier, and speed up our move from data to discovery. Our goal is a data-science-dedicated language model that makes high-level analytical tools, leading to a time when the exploration and understanding of data science are no longer held back by the current limits.

Large Language Models (LLMs), a cornerstone in the realm of artificial intelligence, have emerged as a potent tool for storing, manipulating, and inferring from the vast knowledge (Chowdhery et al., 2022; Hoffmann et al., 2022; Huang et al., 2023; Touvron et al., 2023a, OpenAI, 2023). Despite their broad potential, these models struggle with limitations, notably a deficiency in expert knowledge across specialized domain. This limitation has led to the pursuit of enhancing LLMs for specific applications like science has led to significant research, indicating substantial room for improvement and refinement (Taylor et al., 2022). Similar to this study, the aim of our research is to enable LMs to effectively learn domain-specific knowledge in the domain of data science.

Question Answering (QA) is a crucial task to test an LLM's domain-specific knowledge. However, QA systems have shown remarkable proficiency in processing complex natural language contexts, encapsulating a wide array of implicit textual knowledge (Liu et al., 2019; Lee et al., 2019; Liu et al., 2021). Various prompting methods have been proposed to overcome the challenges associated with answering complex questions. Approaches like Chain-of-Thought and Tree-of-Thought have guided LLMs to solve more complex problems through step-by-step reasoning (Wei et al., 2023; Yao et al., 2023). More recently, a prompting method called Graph-of-Thoughts has been suggested, which designs prompts resembling graph structures, similar to human thought processes and brain mechanisms (Besta et al., 2023a). Meanwhile, numerous studies have explored the integration of Knowledge Graphs (KGs) with QA systems, aiming to enhance structured reasoning capabilities (Mihaylov and Frank, 2018; Lin et al., 2019; Feng et al., 2020; Wang et al., 2021; Yasunaga et al., 2022).

Our research takes a broader approach to address these challenges in LLMs. In contrast to solely relying on previous methodologies, we explore various fine-tuning methods that enrich LLMs with data science expertise. This includes not only utilizing KGs but also experimenting with diverse prompting techniques and dedicated training regimes to augment the models' comprehension and reasoning abilities across the spectrum of data science questions.

Our research is driven by a twofold objective: firstly, to develop and refine domain-specific fine-tuning methodologies for LMs in data science, ensuring optimized responses to specialized queries; and secondly, to enable users to effectively acquire and apply data science knowledge through interactions with these fine-tuned models.

Zhou et al. (2023) have demonstrated that fine-tuning with quality-focused datasets yields better results than using large-scale, indiscriminately collected data. However, smaller LLMs continue to struggle with issues such as prompt misunderstanding and hallucinations. Our proposed GRAPHENE framework seeks to address these challenges by employing tailored instruction and system tokens, as well as various prompting techniques, making it suitable for both subjective and multiple-choice QA tasks. It incorporates a parameter-efficient graph neural network for enhanced reasoning in multiple-choice contexts (Hendrycks et al., 2021).

Our **contributions** are as follows:

- Maximizing small-scale LLM performance in data science QA tasks, thereby pushing the boundaries of what these models can achieve in specialized domains.
- Introducing graph-derived prompts, a novel approach that enhances the LLM's understanding and interaction with complex queries.
- Combining knowledge graphs and graph neural networks for a deeply integrated, task-specific system, paving the way for more sophisticated and accurate QA capabilities in a special kind of question in data science.
- Demonstrating significant performance improvements over existing baselines with our GRAPHENE approach, establishing a new benchmark for LLM applications in the field of data science (Touvron et al., 2023b).

This paper details our methodologies, the insights gained, and the implications of our findings, setting the stage for a new era in intelligent, domain-specific question answering systems, especially in the data science domain.

2 Background and Related Work

The development and fine-tuning of language models (LMs) have been pivotal in advancing Question Answering (QA) systems and natural language processing. Our study reviews key contributions and methodologies in this evolving field.

Llama2 serves as a foundational model and a benchmark in the realm of open foundation and fine-tuned chat models. It represents the baseline against which we compare advancements in language modeling (Touvron et al., 2023b). This approach is complemented by the insights from Zhou et al. (2023), which underscores the effectiveness of training models with smaller, systematically chosen datasets. In particular, we demonstrate improved performance with 1000 well-selected training prompts in areas such as computer science, statistics, and AI, compared to training with large, indiscriminate datasets.

Taylor et al. (2022) exemplifies the success of domain-specific fine-tuning in LLMs, highlighting the potential of specialized models in scientific domains. This approach showcases how targeted fine-tuning can lead to significant advancements in domain-specific applications. In a similar vein, studies like Park et al. (2023) and Yang et al. (2023) have shown the benefits of integrating language models with other modalities, such as embedding, to enhance QA capabilities. This integration highlights the recent successes of multi-modal models and suggests that incorporating our focus on graph modality into language models holds promise for enhancing complex reasoning tasks. This approach aligns with the evolving landscape of multi-modal language models, positioning our work at the forefront of innovative and sophisticated AI systems.

In the area of prompt tuning, works like Tian et al. (2023), Liu et al. (2023b), Wen et al. (2023), and Ye et al. (2023) focus on incorporating knowledge graph structures. These approaches suggest that graph-based structures can significantly enhance the reasoning capabilities of LMs by providing structured, contextual information. Complementing these techniques are advanced prompting methods like Wei et al. (2023), Yao et al. (2023), and Besta et al. (2023b). These methods have been developed to enable LLMs to mimic human-like reasoning processes, facilitating more effective problem-solving and decision-making capabilities in AI systems.

Finally, the efficiency of LLMs in resource-constrained environments is a critical area of research. Studies like Wu et al. (2020), Hu et al. (2021), and Liu et al. (2022) discuss methods for fine-tuning LLMs within limited resources. These techniques are crucial for model deployment and adaptation in practical settings. In terms of evaluation, resources like Sai et al. (2020) and Liu et al. (2023a), including insights from OpenAI (2023), provide crucial metrics for assessing LMs' performance in generating human-like text.

Collectively, these studies inform our approach, emphasizing the importance of efficient fine-tuning, domain-specific adaptations, and innovative prompting techniques in enhancing the capabilities of language models for data science applications.

3 Method

In our approach, distinct methodologies were applied for open question-answering and multiple-choice question-answering tasks, reflecting the unique requirements of each. The details of the models used for each methodology, along with their respective prompts and hyperparameters, are comprehensively documented in the appendix. This discrimination ensures a focused and detailed presentation of the methodologies, allowing for clear replication and understanding of our approach.

3.1 Baseline

For our baseline, we employed the 7B version of Meta AI's Llama2 Touvron et al. (2023b). Llama2, known for its impressive scalability and adaptability across various natural language processing tasks, offers a robust framework for performing question answering and text generation. This choice aligns with our objective to establish a solid reference point for comparing performance, analyzing the impact of our GRAPHENE methodology, and demonstrating the capabilities of fine-tuned LLMs in domain-specific data science applications.

3.2 Open Question Answering with Hand-Crafted Data

3.2.1 Data Collection

Informed by the insights from Zhou et al. (2023), our data collection process emphasized quality over quantity in the context of fine-tuning. We selected a demanding collection approach, diverging from the standard practice of indiscriminate internet crawling, to gather high-quality data specific to the field of data science.

Our dataset primarily comprises interview questions and answers, forming the basis of our Question Answering Task. Selected with guidance from insights by Kashani and Ivry (2021) on key areas of AI interviews, these questions were chosen specifically because they encapsulate the core concepts of data science, ensuring the relevance and depth of the dataset. This approach aligns to enable users to acquire data science knowledge through conversational interactions, simulating real-world interview scenarios.

The dataset was categorized into three primary domains to encompass diverse and critical areas in data science. In the **Computer Science** domain, we included 500 questions covering topics such as computer architecture, operating systems, and networks. For **Statistics**, another set of 500 questions was compiled, focusing on probability theory, mathematical statistics, and regression analysis. The **AI** category comprised 500 questions that delved into the fundamentals of machine learning, deep learning, computer vision, and natural language processing.

We paid special attention to the accuracy of mathematical expressions and data representations. This was crucial to ensure their suitability for a language format and the overall quality of the dataset. Ultimately, our dataset contained 1481 questions, of which 1000 were designated for the training set and 481 for the testing set. This division was strategically planned to provide a balanced mix for a

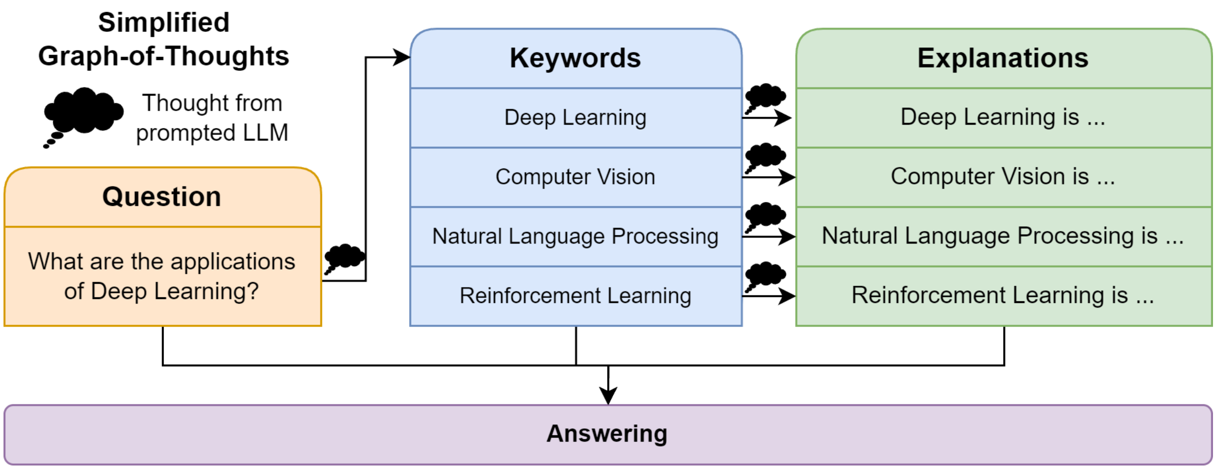

Figure 1: Overview of Simplified Graph-of-Thoughts approach. We ask LLMs to identify essential keywords and explain each, integrating them with the original question for a comprehensive answer.

comprehensive assessment of the model's performance across the various topics.

3.2.2 Fine-tuning

In this section, we fine-tuned the baseline model using simple descriptions along with questions and answers as prompts, based on the Open Question Answering dataset, as mentioned previously.

3.2.3 Instruction Prompting and Tuning

Research as FLAN demonstrate that using instruction-style prompts can enhance zero-shot performance, and instruction prompt tuning yields better results in QA tasks (Wei et al., 2021). Thus, for the QA task, we employed specifically structured prompts that separated clear instruction segments alongside separate question and answer components. This approach aimed to leverage the model's capacity to parse and comprehend structured inputs, thereby facilitating more accurate responses.

Instruction prompting Initially, the baseline model was given with only the instruction prompts to gauge its raw inferential capabilities.

Instruction tuning Following this preliminary assessment, we implemented fine-tuning on the training dataset with the addition of guidance from the instruction prompts, which we hypothesized would align the model's learning process more closely with the task-specific requirements, thereby optimizing performance during subsequent inferences.

3.2.4 Simplified Graph-of-Thought

Simplified Graph-of-Thought method builds upon the Graph of Thoughts (GoT) framework, an innovative approach that significantly broadens the capability of large language models (LLMs) to process and generate responses (Wei et al., 2023). Unlike previous methods like Chain-of-Thought or the more rudimentary Tree of Thoughts (ToT), GoT is inspired by the intricate connections found in human thinking or brain networks. The primary strength of GoT is its flexible graph-based structure which mirrors the way our brains organize and connect pieces of information. In this structure, individual thoughts generated by an LLM are viewed as nodes, with edges representing the relationships between these nodes. This allows for a thoughtful combination of information and the use of loops to refine thoughts.

Despite its sophistication, GoT's complex structure can be a hurdle for practical application. Taking cues from Hulbert (2023), who effectively simplified the conceptual elements of the ToT framework for easy prompting, we tailored the prompt of GoT to simplify GoT's methodology:

1. Request LLMs to figure out core keywords from the initial query (prompt 1).

2. Request LLMs to provide straightforward explanations for each keyword generated by LLMs (prompt 2).

3. Combine the original question with the keywords and their explanations to form a answer (prompt 3).

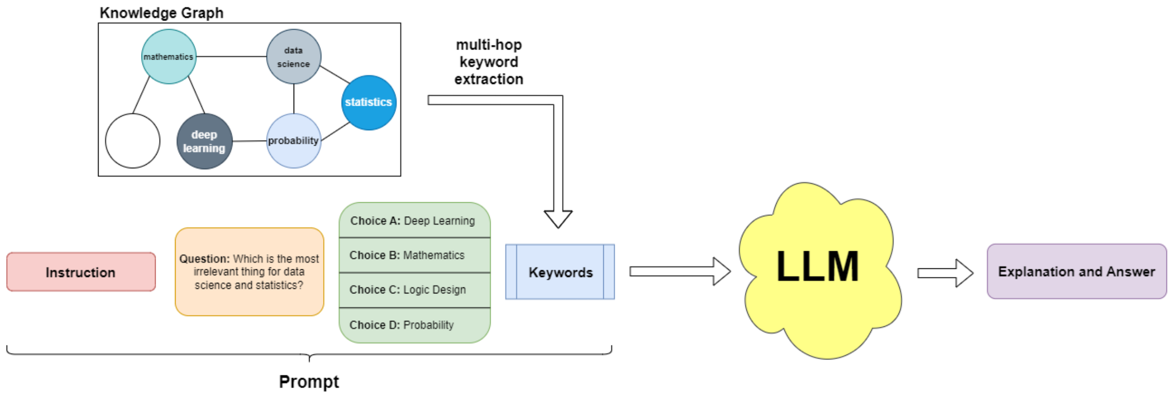

Figure 2: Overview of KG instruction approach. Given a large knowledge graph including tokens of whole questions, we extract keywords from the current question. All of the information obtained in this way is delivered to LLM and then encouraged to explain it.

3.3 Multiple Choice Question Answering with MMLU Dataset

3.3.1 Fine-tuning

During our fine-tuning process, we observed a tendency for the model to generate new questions and options repetitively, rather than answering the given questions, when using identical training and prompt formats. This phenomenon appeared to be a result of the model inadvertently memorizing the format of the training data, leading to unproductive outputs. To mitigate this, we explored the addition of an example in the prompt, akin to employing few-shot learning techniques, even with a model that has already undergone fine-tuning. This strategy seemed particularly effective in the context of smaller-scale LLMs, where combining fine-tuning with few-shot prompting yielded favorable results. Additionally, hyperparameter adjustments were made to optimize this process. For a detailed account of these modifications, readers are referred to the appendix, where we have provided a comprehensive breakdown of the changes implemented.

3.3.2 Fine-tuning with Chain-of-Thought

We draw upon the recently introduced Graph of Thoughts (GoT) framework, a method that substantially enhances the prompting capabilities of large language models (LLMs) beyond existing paradigms like Chain-of-Thought or Tree of Thoughts (ToT).

Building upon the fine-tuning methodology mentioned in the preceding section, we introduce additional insights to the model. According to the methodology used in Wei et al. (2023), employing Chain-of-Thought (CoT) is effective in few-shot or zero-shot tasks. However, for smaller-scale LLMs, we found that a combination of fine-tuning for a limited number of epochs followed by adopting a CoT prompt format is also beneficial. Unlike large-scale models such as GPT-3.5 and GPT-4 (Ouyang et al., 2022; OpenAI, 2023), our baseline model has a smaller scale, which inherently limits the amount of fundamental information it possesses. To address this limitation, we first employed fine-tuning techniques to familiarize the model with the type of problems we aim to solve. Subsequently, we utilized the CoT approach to maximize the extraction of information from the model. This method proved particularly effective in leveraging the available data and enhancing the model's problem-solving abilities.

3.3.3 Knowledge Graph Instruction

Huang et al. (2023) introduces a prompting method designed to address node classification, a common task in Graph Neural Networks (GNNs). While initially seeming misaligned with our focus on natural language generation, this approach finds applicability in multiple-choice question answering (MCQA), which is a kind of classification task. A key distinction between node classification and our task lies in utilizing specialized knowledge more extensively. Based on this understanding, we developed a methodology applicable not just to common graphs but specifically to knowledge graphs.

For our prompt construction, we developed a two-step process: firstly, extracting the most important keywords from the question, and secondly, providing information about the neighbors of these extracted keywords from a pre-constructed knowledge graph. The criterion for "important keywords"

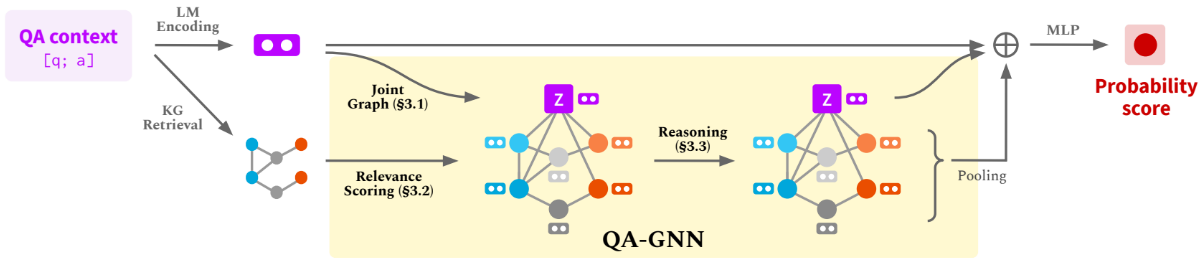

Figure 3: Overview of graph reasoning layer apporch. Given a QA context (**Z**), we connect it with the retrieved KG to form a joint graph, compute the relevance of each KG node conditioned on **Z**; node shading indicates the relevance score, and perform reasoning on the working graph. This figure is from Yasunaga et al. (2022).

is based on tf-idf scores. We extract words within an n-hop distance that are top-k_1 in terms of tf-idf from the words appearing in the question. Among these extracted words, again the top-k_2 words with the highest tf-idf scores are selected. In this process, we exclude words representing integers, reals, `sqrt`, `frac`, and numeric variables, to maintain the focus on relevant keywords.

We constructed a comprehensive knowledge graph by tokenizing all questions. The keywords for each problem we aimed to solve were then identified by the current question token and extracting relevant keywords from this knowledge graph. Notably, words from the choices or answers were not included in the knowledge graph. This decision was made to prevent the model from focusing on words from incorrect choices.

3.3.4 Detour: Graph Reasoning Layer

In a departure from previous methods, which primarily focused on fine-tuning LLMs or providing novel prompts, we explored the utilization of more classical deep learning techniques to tackle our tasks. This approach integrates the pre-trained language model embeddings from RoBERTa (Liu et al., 2019) with knowledge graph embeddings, building up to inference through a traditional graph neural network. The overall architecture of our model draws inspiration from the structure of Yasunaga et al. (2022), ensuring a robust framework for reasoning and inference.

Our knowledge graph is constructed based on the extensive information from ConceptNet (Speer et al., 2018), augmented with specialized knowledge from the MMLU dataset (Hendrycks et al., 2021). The nodes in our knowledge graph are categorized into four types: **Q** nodes representing questions, **A** nodes for answers, **O** nodes for other elements, and **Z** nodes that serve as connectors. Notably,

those in mathematics or statistics, the intersection of **Q** and **A** nodes frequently results in an empty set. While it's impractical to add every numeral from the MMLU dataset into the knowledge graph due to clear limitations, we introduce special tokens as a workaround. For instance, the special token `[1]` indicates a computational problem, encouraging the model to interpret and solve it accordingly.

During the training process, ConceptNet's embeddings are frozen to maintain their integrity and ensure that the model relies on a stable base of knowledge while learning to reason over the graph structure. This hybrid approach, combining the strengths of pre-trained language models with classical graph reasoning, represents our innovative stride towards more effective and nuanced problem-solving in our tasks.

4 Experiments

Our evaluation of the GRAPHENE methodology was conducted over two distinct datasets. The first dataset consists of hand-crafted subjective interview questions, specifically designed to probe the depth and applicability of data science knowledge. For a detailed description of this dataset, readers are referred to section 3.2.1.

The second dataset represents a restructured version of the MMLU dataset (Hendrycks et al., 2021). From the broad array of topics covered in the MMLU benchmark, we selectively harvested questions pertinent to the field of data science. This restructured dataset was then divided into a new training set and testing set, tailored to assess the specific competencies and knowledge areas relevant to data science.

Instead of a separate ablation study section, our experimental design inherently incorporates the ablation approach. We incrementally added and re-

Model	GPT-4 Score	METEOR	NIST
Llama2 7B (Baseline)	3.922	0.2620	1.1904
Llama2 7B + Fine-Tuning	3.664	0.2640	**1.4353**
Llama2 7B + Instruction Prompting	3.923	0.2548	1.0095
Llama2 7B + Instruction Tuning	**4.142**	**0.2675**	1.1812
Llama2 7B + Simplified GoT	3.288	0.2383	1.3045

Table 1: Comparison of task 1 performance in terms of GPT-4 Score, METEOR, and NIST metrics.

Model	Accuracy (%)	AGR (%)
Llama2 7B (Baseline)	7.32	21.46
Llama2 7B + Fine-Tuning	18.05	67.80
Llama2 7B + CoT	28.29	92.19
Llama2 7B + KG Instruction	28.78	94.15
RoBERTa + KG + GNN	**32.68**	**100.00**
GPT-3.5	36.00	100.00
GPT-4	65.37	100.00

Table 2: Comparison of task 2 performance in terms of Accuracy and AGR.

fined ideas, starting from the baseline and advancing through successive iterations. This progression is embedded within the experimental narrative, as each step builds upon the last, eliminating the need for an isolated ablation study. Such a format allows us to demonstrate the effectiveness of each component within the holistic context of the GRAPHENE approach.

By employing these two datasets, we aimed to provide a comprehensive evaluation of the GRAPHENE approach. The first dataset, consisting of open questions from interview scenarios, challenges the model's capacity to process and respond to intricate queries with definitive solutions, mirroring the depth and precision required in professional settings (**Task 1**). Equally important is the second dataset, derived from MMLU, which tests the model's ability to swiftly navigate multiple-choice questions, emphasizing accurate response selection (**Task 2**).

4.1 Metrics

In **Task 1**, we assess the performance of above generative question answering models using a set of metrics designed to measure their effectiveness in producing accurate, coherent, and contextually appropriate answers. Specifically, we employ three metrics, referring to the comprehensive survey on Sai et al. (2020): METEOR (Banerjee and Lavie, 2005), NIST (Doddington, 2002), and G-Eval (Liu et al., 2023a).

METEOR is a metric that addresses several of the weaknesses found in the BLEU metric. It is designed to better reflect human judgment by not only identifying exact word matches between the generated output and reference translations but also by recognizing synonyms and paraphrases, making it a suitable choice for evaluating the nuanced task of generative question answering. Furthermore, METEOR incorporates a stronger emphasis on recall. This makes it particularly valuable for ensuring that the generated answers capture the necessary information requested in the questions.

On the other hand, **NIST**, while also building upon the foundations of BLEU, refines the concept of n-gram precision. Unlike BLEU, NIST calculates the arithmetic mean, consequently allocating different weights to n-grams of various lengths. Moreover, NIST introduces an improved brevity penalty that mitigates the issue of favoring overly short translations, which can be especially critical in the context of generating complete and informative answers in a question-answering system.

Along with traditional machine translation evaluation metrics such as METEOR and NIST-MT, our evaluation methodology also incorporates a novel scoring system known as G-Eval, utilizing the capabilities of the state-of-the-art language model, GPT-4.

G-Eval (GPT-4 Score), leverages the advanced language understanding and generation capabilities of GPT-4, which is recognized for its superior performance in a range of natural language processing tasks. Emerging research indicates that

GPT-4's assessments are highly correlational with human judgments, often outstripping other SOTA (state-of-the-art) evaluators in terms of precision and relevance.

For scoring, the process is conducted through an interactive routine where GPT-4 is presented with prompts that comprise both the questions and their corresponding generated answers. It is then prompted to rate the quality of the answers on a numerical scale from 1 to 5, with 1 being the lowest and 5 representing the highest quality. This human-like scoring approach provides a nuanced interpretation of the model's output, assessing not just factual accuracy but also considering the naturalness and coherence. Detailed prompts and scoring methodology for G-Eval can be found in the Appendix.

In evaluating **Task 2**, we employed two primary metrics to comprehensively assess the performance of the language model.

The first metric is **accuracy**, which is vital in understanding how often the language model selects the correct answer from options A, B, C, or D. In instances where the LLM's output is truncated before it generates a final answer due to maximum token limitations, or if it diverts from providing an answer altogether, we classify such instances as option E. Thus, the evaluation is conducted as a 5-option multiple-choice question, incorporating this additional consideration.

Simultaneously, we introduce the **Answer Generation Ratio (AGR)**, a metric designed to quantify the proportion of instances where the LLM successfully generates a response that can be categorized as one of the options A, B, C, or D. This metric provides insight into the model's ability to not just provide any answer, but to generate one that aligns with the structured format of the given options. To ensure a fair assessment, even in the methodology described in section 3.3.4, the dimensionality of the output from the final linear layer is set to 5.

Together, these metrics—accuracy and AGR—offer a nuanced view of the model's performance, capturing both its ability to select the correct answer and its effectiveness in generating coherent, option-aligned responses.

4.2 Experimantal Setup

Our experimental framework was structured to evaluate two distinct tasks, each with their own specific setup to ensure optimal performance and accurate results. Details on the hyperparameter settings for each task have been comprehensively documented in the appendix.

Task 1 was conducted using Colab Pro+ with the Llama2 7B model. We employed a 4-bit quantization and 16-bit floating point precision, complemented by techniques like LoRA and PEFT for efficient fine-tuning. The training phase was executed on an A100 40GB GPU, while the generation phase utilized a T4 GPU to balance computational efficiency and cost.

For **Task 2**, we maintained a similar environment on Colab Pro+ with Llama2 7B, using 4-bit quantization and 16-bit floating point precision. The training was once again performed on an A100 40GB GPU. However, the knowledge graph construction and generation phases were carried out on a T4 GPU, reflecting the task's differing computational demands.

In the graph reasoning layer methodology, knowledge graph construction leveraged the A100 40GB GPU's robust processing capabilities. Subsequently, the training and reasoning steps were conducted on a T4 GPU, optimizing the resource allocation for the task's requirements.

4.3 Main Results

4.3.1 Open Question Answering

Fine-tuning In the experiment displayed on the second row of Table 1, it was noted that the GPT-4 Score for the fine-tuned model fell short of that exhibited by the baseline model. Upon further analysis, it was observed that when the fine-tuned model was provided with the same prompt as the baseline, it had a tendency for creating additional questions and answers instead of directly addressing the presented question. This led to a recursive pattern of responses which, proved counterproductive. This behavior appears to be the primary factor contributing to the reduced GPT-4 Score observed in the fine-tuned model. However, an increase in both the METEOR and NIST-MT metrics indicates successful learning with respect to the dataset.

Instruction Prompting Observing the third row of Table 1, it is evident that there was no significant change in the metrics. The GPT-4 Score remains almost identical, which suggests that merely providing instruction prompts did not assist the baseline model in achieving better clarity or explanation.

Instruction Tuning Upon examining the fourth row of Table 1, a substantial increase is observed in both the GPT-4 Score and the METEOR metric. This is in clear contrast to the previous result where instruction prompting exhibited negligible changes. These results underscore the effectiveness of instruction tuning in QA tasks, highlighting its potential for significantly improving performance metrics.

Simplified Graph-of-Thought Reflecting upon the fifth row of Table 1, there was a significant drop in both the GPT-4 Score and the METEOR metric. Further analysis revealed a couple of issues. Initially, the process to generate keywords produced several unnecessary ones. Moreover, trying to incorporate questions, keywords, and explanations within a single prompt led to excessive length, which consequently resulted in the model's failure to grasp the intention of the given question and provide a clear answer. These complications likely contributed to the diminishment observed in the GPT-4 Score and METEOR metric. On the contrary, the NIST metric showed improvement. It is presumed that despite not capturing the precise intentention of the questions, the model was able to generate responses that were similar with the questions, which accounts for the increase in the NIST metric.

The simplification that involved omitting the Scoring module used in the Graph-of-Thought approach seems to have led to an issue where the generated thoughts were neither evaluated nor filtered. It appears that the simplified GoT method presents opportunities for further refinement.

4.3.2 Multiple Choice Question Answering

The same problem occurred in the **fine-tuning** model of Open Question Answering - generating new questions and options rather than answering the given question. To prevent this, we incorporated an additional example into the prompt, emulating a few-shot learning approach even with the already fine-tuned model. For small-scale LLMs, combining fine-tuning with few-shot prompts proved to be an effective strategy.

The effectiveness of combining **fine-tuning with Chain-of-Thought** (CoT), as represented in the third row of Table 2, was inspired by the shortcomings of the aforementioned approach. While large-scale LLMs typically utilize CoT in few-shot or zero-shot scenarios, our experiments with smaller LLMs demonstrated that after fine-tuning, the concurrent use of CoT can extract maximum information within a limited scale. As detailed in the appendix, the conventional zero-shot CoT prompt "Let's think step-by-step." was adapted for a 1-shot scenario, which significantly contributed to the method's success.

For the experiment listed in the fourth row of Table 2, i.e. **KG Instruction**, we selected top-k_1 words based on TF-IDF scores and used the top-k_2 words within an n-hop distance. The parameters were set to $k_1 = 20, k_2 = 20, n = 3$. Excessive k_1 or n values led to an over-representation of certain words as keywords, whereas a very small k_2 resulted in insufficient information being provided to the model.

In the fifth row of Table 2, i.e. **Graph Reasoning Layer** method, we utilized fixed ConceptNet embeddings alongside a trainable basic GNN with 5 layers (Veličković et al., 2018). See Appendix for details on the GNN architecture. After the train-validation split, we performed data augmentation on the training data, generating five variations for each question by substituting one word with a synonym, resulting in a sixfold increase in training data. To prevent overfitting, we set a dropout rate of 0.5.

Our findings indicate that for multiple-choice questions, effective learning can be achieved with a well-defined set of about 1000 data points. For small-scale LLMs, a combined approach of fine-tuning and prompt tuning emerges as a viable method. Additionally, leveraging pre-trained language models (PLMs) shows superior performance compared to smaller-scale LLMs.

5 Conclusion

In conclusion, our study demonstrates that the GRAPHENE methodology—leveraging graph-based analysis and prediction with efficient natural language exploration—substantially improves data science QA tasks. By innovatively integrating fine-tuning methodologies and prompting techniques, we not only advance the capabilities of smaller-scale LLMs but also set a new benchmark for specialized domain applications, paving the way for future enhancements in intelligent, domain-specific question answering systems.

In future work, critical questions must be addressed to refine and advance our GRAPHENE methodology. Notably, the reliability of GPT-4's scoring system warrants investigation, especially considering its tendency to favor lengthier responses, which might skew the evaluation of more concise but equally valid answers. Additionally, the exploration of a more sophisticated and detailed Simplified Graph-of-Thoughts methodology promises to enhance the depth and accuracy of our model's reasoning capabilities. Finally, the advent of post-LLaMA LLMs presents an exciting opportunity to further improve performance, suggesting that incorporating newer and more advanced language models could significantly bolster our framework's effectiveness in the ever-evolving field of data science.

Acknowledgements

First, the authors thank Gyeongmin Kim for his dedicated teaching and constructive advice on the project's development. Additionally, the authors also thank Minseong Bae of Korea University for his role as a reviewer and for providing valuable resources.

References

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization, pages 65–72.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2023a. Graph of thoughts: Solving elaborate problems with large language models.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. 2023b. Graph of thoughts: Solving elaborate problems with large language models. arXiv preprint arXiv:2308.09687.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In Proceedings of the second international conference on Human Language Technology Research, pages 138–145.

Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. 2020. Scalable multi-hop relational reasoning for knowledge-aware question answering.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models.

Shaohan Huang, Li Dong, Wenhui Wang, Yaru Hao, Saksham Singhal, Shuming Ma, Tengchao Lv, Lei Cui, Owais Khan Mohammed, Barun Patra, Qiang Liu, Kriti Aggarwal, Zewen Chi, Johan Bjorck, Vishrav Chaudhary, Subhojit Som, Xia Song, and Furu Wei. 2023. Language is not all you need: Aligning perception with language models.

Dave Hulbert. 2023. Using tree-of-thought prompting to boost chatgpt's reasoning. https://github.com/dave1010/tree-of-thought-prompting.

Shlomo Kashani and Amir Ivry. 2021. Deep learning interviews: Hundreds of fully solved job interview questions from a wide range of key topics in ai. arXiv preprint arXiv:2201.00650.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. Bioinformatics, 36(4):1234–1240.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. Kagnet: Knowledge-aware graph networks for commonsense reasoning.

Fangyu Liu, Ehsan Shareghi, Zaiqiao Meng, Marco Basaldella, and Nigel Collier. 2021. Self-alignment pretraining for biomedical entity representations.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023a. G-eval: Nlg evaluation using gpt-4 with better human alignment.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Zemin Liu, Xingtong Yu, Yuan Fang, and Xinming Zhang. 2023b. Graphprompt: Unifying pre-training and downstream tasks for graph neural networks.

Todor Mihaylov and Anette Frank. 2018. Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge.

OpenAI. 2023. Gpt-4 technical report.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Jinyoung Park, Hyeong Kyu Choi, Juyeon Ko, Hyeonjin Park, Ji-Hoon Kim, Jisu Jeong, Kyungmin Kim, and Hyunwoo J. Kim. 2023. Relation-aware language-graph transformer for question answering.

Ananya B. Sai, Akash Kumar Mohankumar, and Mitesh M. Khapra. 2020. A survey of evaluation metrics used for nlg systems.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2018. Conceptnet 5.5: An open multilingual graph of general knowledge.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V. Chawla, and Panpan Xu. 2023. Graph neural prompting with large language models.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks.

Kuan Wang, Yuyu Zhang, Diyi Yang, Le Song, and Tao Qin. 2021. Gnn is a counter? revisiting gnn for question answering.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. arXiv preprint arXiv:2109.01652.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Yilin Wen, Zifeng Wang, and Jimeng Sun. 2023. Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models.

Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. 2020. Integer quantization for deep learning inference: Principles and empirical evaluation.

Yue Yang, Kaipeng Zhang, Yuying Ge, Wenqi Shao, Zeyue Xue, Yu Qiao, and Ping Luo. 2023. Align, adapt and inject: Sound-guided unified image generation.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. arXiv preprint arXiv:2305.10601.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2022. Qa-gnn: Reasoning with language models and knowledge graphs for question answering.

Ruosong Ye, Caiqi Zhang, Runhui Wang, Shuyuan Xu, and Yongfeng Zhang. 2023. Natural language is all a graph needs.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment.

A Used Prompts

In this appendix, we introduce the prompts used in fine-tuning the models.

A.1 Open Question Answering with Hand-Crafted Data

A.1.1 Fine-tuning

You are applying for a job related to AI, and you can expect to encounter problems related to statistics, computer science, and artificial intelligence.

Question:
`{Question}`

Answer and Explanation:
`{Answer}`

A.1.2 Instruction and Instruction-Tuning

During inference, the `{{Answer}}` was excluded.

Below is an instruction that describes a task. Write a response that appropriately completes the request.
`[INST] <<SYS>>`
You are a job applicant with expertise in Mathematics, Statistics, Computer Science, and Machine Learning.
Please write the answer of the following interview question with a clear and comprehensive explanation to demonstrate depth of knowledge.
Please answer only the given one question without any additional information.
`<</SYS>>`
Question:
`{{Question}} [/INST]`
Answer:
`{{Answer}}`

A.1.3 Simplified Graph-of-Thought

Prompt 1

To better generate keywords, a one-shot example has been added in prompt as follows.

Below is an instruction that describes a task. Write a response that appropriately completes the request.
[INST] <<SYS>>
You have expertise in Mathematics, Statistics, Computer Science, and Machine Learning.
Please write the keywords of the following interview question to use your knowledge.
<</SYS>>

Example:
Question: What are the applications of Deep Learning?
Keywords of question:
Deep Learning, Computer vision, Natural language processing (NLP), Reinforcement learning

Referencing the example above, find the key keywords in the below question.
Question: {{Question}} [/INST]

Keywords of question:

Prompt 2

Below is an instruction that describes a task. Write a response that appropriately completes the request.
[INST] <<SYS>>
You have expertise in Mathematics, Statistics, Computer Science, and Machine Learning.
Please provide a simple but clear explanation for each of the keywords below to demonstrate depth of knowledge.
<</SYS>>

Keywords: {{Keywords}} [/INST]
Explanation of keywords:

Prompt 3

Below is an instruction that describes a task. Write a response that appropriately completes the request.
[INST] <<SYS>>
You are a job applicant with expertise in Mathematics, Statistics, Computer Science, and Machine Learning.
Please write the answer of the following interview question with a clear and comprehensive explanation to demonstrate depth of knowledge.
Please answer only the given one question without any additional information.

Question: {{Question}}
Keywords of question: {{Keywords}}
Explanations of keywords:
{{Explanations}}
<</SYS>>
Only refer to keywords or explanations when necessary; focus on providing an accurate answer to the given question.

[/INST]
Answer:

A.2 Multiple Choice Question Answering with MMLU Dataset

A.2.1 Baseline

You are a job applicant in an interview, tackling questions that fall into one of the following categories: Mathematics, Statistics, Computer Science, or Machine Learning.
For each question, you are presented with four options: A, B, C, and D.
You must choose only one correct answer from these options, without any additional explanation or reasoning.

Question: `{eval_entry['Question']}`

A: `{eval_entry['A']}`
B: `{eval_entry['B']}`
C: `{eval_entry['C']}`
D: `{eval_entry['D']}`

Answer:

A.2.2 Fine-tuning

You are a job applicant in an interview, tackling questions that fall into one of the following categories: Mathematics, Statistics, Computer Science, or Machine Learning.
For each question, you are presented with four options: A, B, C, and D.
Your sole task is to select one correct answer from these options for the provided questions, without creating new questions or providing additional explanations.

Example Question: In machine learning, what does the term 'gradient descent' refer to?

A: A method for dividing data into clusters
B: An algorithm for optimizing a function
C: A technique for classifying data
D: A type of neural network architecture

Answer: B

Question: `{eval_entry['Question']}`

A: `{eval_entry['A']}`
B: `{eval_entry['B']}`
C: `{eval_entry['C']}`
D: `{eval_entry['D']}`

Answer:

A.2.3 Fine-tuning with Chain-of-Thought

You are a job applicant in an interview, tackling questions that fall into one of the following categories: Mathematics, Statistics, Computer Science, or Machine Learning.
For each question, you are presented with four options: A, B, C, and D.
Your sole task is to select one correct answer from these options for the provided questions, without creating new questions or providing additional explanations.

Example Question: In machine learning, what does the term 'gradient descent' refer to?

A: A method for dividing data into clusters
B: An algorithm for optimizing a function
C: A technique for classifying data
D: A type of neural network architecture

'Gradient descent' is a term often used in optimization problems.
It involves iteratively moving towards the minimum of a function.
This process is not specific to dividing data, classifying data, or a neural network architecture.
Therefore, it aligns best with the concept of an algorithm for optimizing a function.

Answer: B

Question: `{eval_entry['Question']}`

A: `{eval_entry['A']}`
B: `{eval_entry['B']}`
C: `{eval_entry['C']}`
D: `{eval_entry['D']}`

Let's think step-by-step.

A.2.4 Knowledge Graph Instruction

You are a job applicant in an interview, tackling questions that fall into one of the following categories: Mathematics, Statistics, Computer Science, or Machine Learning.
For each question, you are presented with four options: A, B, C, and D.
Your sole task is to select one correct answer from these options for the provided questions, without creating new questions or providing additional explanations.
Additionally, use the given keywords to provide the explanation behind your answer selection.

Question: `{eval_entry['Question']}`

A: `{eval_entry['A']}`
B: `{eval_entry['B']}`
C: `{eval_entry['C']}`
D: `{eval_entry['D']}`

Keywords:
`{', '.join(final_keywords)}`

Explanation:

B Hyperparameter settings

We used `BitsAndBytesConfig` and `trainer` of the huggingface transformers when loading and fine-tuning the models. We also used the `generate` of the PeftModel from peft when creating the text.

B.1 Open Question Answering with Hand-Crafted Data

B.1.1 Model Loading Configuration

Hyperparameter	Value
Load in 4-bit	True
Use double quantization	True
Quantization type	`nf4`
Compute dtype	`torch.bfloat16`
pretraining_tp	1

Table 3: Hyperparameters of `BitsAndBytesConfig` used for loading the Llama2 7B.

B.1.2 Fine-tuning

Hyperparameter	Value
Number of epochs	15
Batch size	2
Gradient accumulation steps	16
Learning rate	1e-5
Mixed precision training	True (A100)
Optimizer	`paged_adamw_8bit`
Weight decay	0.1
Learning rate scheduler	linear
Warm-up steps	0

Table 4: Hyperparameters of `trainer` used for fine-tuning the Llama2 7B.

Hyperparameter	Value
Max new tokens	512
Number of return sequences	1

Table 5: Hyperparameters of `generate` settings for fine-tuning the Llama2 7B.

B.1.3 Instruction-Tuning

Hyperparameter	Value
lora_alpha	16
lora_dropout	0.1
r	64
bias	none
task_type	`CASUAL_LM`

Table 6: Hyperparameters of `LoraConfig` settings for fine-tuning with Chain-of-Thoughts approach.

Hyperparameter	Value
Number of epochs	15
Batch size	2
Gradient accumulation steps	16
Learning rate	1e-5
Mixed precision training	True (A100)
Optimizer	`paged_adamw_8bit`
Weight decay	0.1
Learning rate scheduler	linear
Warm-up steps	0
max_seq_length	512

Table 7: Hyperparameters of `trainer` used for instruction-tuning the Llama2 7B.

Hyperparameter	Value
Max length	1024
Number of return sequences	1

Table 8: Hyperparameters of `generate` settings for instruction-tuning the Llama2 7B.

B.1.4 Simplified Graph-of-Thought

The same hyperparameters used for instruction-tuning have beed applied.

B.2 Multiple Choice Question Answering with MMLU Dataset

B.2.1 Model Loading Configuration

Hyperparameter	Value
Load in 4-bit	True
Use double quantization	True
Quantization type	`nf4`
Compute dtype	`torch.bfloat16`

Table 9: Hyperparameters of `BitsAndBytesConfig` used for loading the Llama2 7B.

B.2.2 Fine-tuning

Hyperparameter	Value
Number of epochs	15
Batch size	2
Gradient accumulation steps	16
Learning rate	2e-5
Mixed precision training	True (A100)
Optimizer	`paged_adamw_8bit`
Weight decay	0.1
Learning rate scheduler	linear
Warm-up steps	0

Table 10: Hyperparameters of `trainer` used for fine-tuning the Llama2 7B.

Hyperparameter	Value
Max new tokens	64
Number of return sequences	1
Temperature	0.7
Top-p	0.9
Top-k	50
Repetition penalty	1.2

Table 11: Hyperparameters of `generate` settings for fine-tuning approach.

B.2.3 Fine-tuning with Chain-of-Thought

Hyperparameter	Value
Max new tokens	256
Number of return sequences	1
Temperature	0.6
Top-p	0.85
Top-k	50
Repetition penalty	1.2

Table 12: Hyperparameters of `generate` settings for fine-tuning with Chain-of-Thoughts approach.

B.2.4 Graph Reasoning Layer

Hyperparameter	Value
Attention head number	2
Batch size	64
Encoder learning rate	1e-05
Decoder learning rate	0.001
Input dropout	0.5
Feature dropout	0.5
Graph dropout	0.5
Encoder	`roberta-large`
Evaluation batch size	2
Fully connected dimension	200
Use FP16	True
Freeze entity embedding	True
GNN dimension	100
Initialization range	0.02
Top-k for evaluation	5
Loss type	Cross Entropy
Learning rate schedule	Fixed
Max gradient norm	1.0
Max node number	200
Max sequence length	100
Mini-batch size	2
Warm-up steps	0
Weight decay	0.01

Table 13: Detailed hyperparameters for graph reasoning layer approach in Section 3.3.4.

C Model Details

In this section, we describe the model architecture of the graph reasoning layer method in Section 3.3.4. Our model leverages a Graph Neural Network (GNN) to effectively address the complex task of text-based question answering. The architecture is composed of several interconnected layers, each serving a distinct purpose in processing and relaying information through the network.

1. **Embedding Layers**: At the core of the model are embedding layers designed to convert discrete data into continuous representations. The `emb_node_type` layer transforms node type information into a dense vector space, providing a nuanced understanding of each node's role within the graph. Concurrently, the `emb_score` layer processes node scores, representing the significance or relevance of each node within the context of a particular query. This dual embedding scheme allows for a rich representation of node characteristics.

2. **Edge Encoding**: The architecture incorporates an `edge_encoder`, a sequential module consisting of linear layers and ReLU activations. This encoder is responsible for translating edge information, including the type of connection and the nature of connected nodes, into a feature-rich representation. By doing so, the model captures the complex interplay between nodes, an essential aspect of understanding graph-structured data.

3. **Graph Attention Convolutional Layers**: A series of `GATConvE` layers form the backbone of the message-passing mechanism. Each layer is designed to propagate and refine node features across the graph, considering both the immediate neighbors and the broader structure. The attention mechanism within these layers allows the model to focus on the most relevant parts of the graph, dynamically adjusting its focus based on the input query and the current state of node features.

4. **Integration and Output Layers**: The `Vh` and `Vx` linear layers, along with a GELU activation function, are used to integrate the various feature representations and produce the final node representations. The model's output is a continuous vector for each node, which can be interpreted or further processed depending on the specific application or task at hand.

This architecture is designed to be flexible and adaptable, capable of handling various types of graph-structured data and queries. Its layered approach, combining embedding, encoding, attention mechanisms, and integration, allows it to capture the complexities of natural language and graph relationships, making it a powerful tool for question-answering systems and other applications in the domain of NLP and graph analytics.

D Qualitative Results

D.1 Open Question Answering with Hand-Crafted Data

D.1.1 Instruction Tuning

`To see the used prompt, go to A.1.2.`

Question: What is a linked list?

Answer:
A linked list is a fundamental data structure in computer science that consists of a sequence of nodes, each of which contains a value and a reference (or pointer) to the next node in the list. The nodes are Ïinkedẗogether through these pointers, allowing for efficient insertion, deletion, and traversal of the list.
A linked list is a type of dynamic memory allocation, where each node represents a block of memory that can be allocated and deallocated as needed. The nodes are typically stored in contiguous memory locations, with each node pointing to the next node in the list. This allows for efficient storage and retrieval of data, as well as efficient manipulation of the list structure itself.
There are several types of linked lists, including:
. . .
Linked lists have several advantages over other data structures, including:
. . .
In conclusion, a linked list is a fundamental data structure in computer science that consists of a sequence of nodes, each of which contains a value and a reference to the next node. Linked lists offer efficient storage and retrieval of data, dynamic memory allocation, and easy implementation, but also have disadvantages such as slow insertion and deletion, memory waste, and limited data structure operations.

D.1.2 Simplified Graph-of-Thoughts

`To see the used prompt, go to A.1.3.`

Question: What is Naive Bayes algorithm, When we can use this algorithm in NLP?

Keywords of question: Naive Bayes algorithm, NLP, Text classification, Spam detection, Sentiment analysis

Explanation of keywords:
1. Naive Bayes algorithm: Naive Bayes is a family of probabilistic supervised learning algorithms that are used for classification tasks. It is called "naive" because it assumes independence between features, even when they are not. Naive Bayes can be used for both binary and multiclass classification tasks.
. . .
5. Sentiment analysis: Sentiment analysis is the task of determining the sentiment of a text document, such as whether it is positive, negative, or neutral. It is a common application of NLP and text classification. Sentiment analysis can be used for marketing research, customer feedback analysis, and other tasks.

Answer: The Naive Bayes algorithm is a popular machine learning algorithm used in Natural Language Processing (NLP) tasks such as text classification, spam detection, and sentiment analysis. It is based on Bayes' theorem and assumes independence between features, which is often untrue in NLP tasks. When we can use the Naive Bayes algorithm in NLP? We can use the Naive Bayes algorithm in NLP when we have a large number of features, and we want to classify text into a few categories. It is a simple algorithm that is easy to implement and computationally efficient. However, the algorithm assumes independence between features, which is often untrue in NLP tasks. Therefore, we may use more advanced techniques such as Logistic Regression or Deep Learning for more complex NLP tasks.

D.1.3 G-Eval

The highest score given (5.0).

Question:
What is POP3?
Answer:
POP3 is a protocol used to retrieve e-mail messages from a remote server.
Explanation: POP3 is a protocol used to retrieve e-mail messages from a remote server. POP3 stands for Post Office Protocol version 3. POP3 allows you to retrieve e-mail messages from a remote server. It is commonly used with email clients like Microsoft Outlook or Mozilla Thunderbird.
In summary, POP3 is a protocol used to retrieve e-mail messages from a remote server.
- Factual Consistency: 5.0

The lowest score given (1.0).

Question:
How can hyperparameters be trained in neural networks?
Answer:
Hyperparameters can be trained in neural networks through the use of techniques such as grid search, random search, or Bayesian optimization. These techniques allow for the exploration of the space of hyperparameters to find the optimal set for a given model architecture.
- Factual Consistency: 1.0

D.2 Multiple Choice Question Answering with MMLU Dataset

D.2.1 Fine-tuning with Chain-of-Thought

`To see the used prompt, go to A.2.3.`

Question: "Magic memory" has two operations: Read and Clear. Both are indivisible and mutually exclusive. Clear sets the magic memory to zero. Read returns a value that represents the number of Read operations since the last Clear operation. Which of the following is (are) true of "Magic memory"?
I. It can provide the functionality of an atomic Test-and-Set.
II. It can be used to coordinate processes running on a shared-memory multiprocessor.
III. It is only useful on a multiprocessor.

A: I only
B: II only
C: III only
D: I and II

Let's think step-by-step.

1. The term "magic memory" implies that the operations Read and Clear have some special property.
2. The statement "Clear sets the magic memory to zero" implies that Clear is both indivisible and mutually exclusive.
3. The statement "Read returns a value representing the number of Read operations since the last Clear operation" implies that Read is also indivisible and mutually exclusive.
4. Since the operations are indivisible and mutually exclusive, they must be performed atomically.
. . .
Therefore, we can conclude that option D is the correct answer.

Answer: D

D.2.2 Knowledge Graph Instruction

`To see the used prompt, go to A.2.4.`

Question: Which of the following points would Bayesians and frequentists disagree on?

A: The use of a non-Gaussian noise model in probabilistic regression.
B: The use of probabilistic modelling for regression.
C: The use of prior distributions on the parameters in a probabilistic model.
D: The use of class priors in Gaussian Discriminant Analysis.

Keywords: bayesians, pentagon

Explanation:
The answer is (C). Both Bayesian and frequentist statisticians employ probabilistic models; however, Bayesians use prior distributions on parameters, whereas frequentists do not. Because Bayesians use prior distributions, they may make different predictions than frequentists when there isn't enough data. Therefore, Bayesians and frequentists disagree on this point.

E Details on G-Eval

E.1 Score function

$$\text{score} = \sum_{i=1}^{n} p(s_i) \times s_i$$

In the process of G-Eval, multiple scores are drawn to mitigate the variability inherent in such evaluative models. In the Score Function, s_i is the score and $p(s_i)$ is the probability of score s_i.

We use a `temperature` setting of 2 and `top_p` of 1 to introduce diversity in the scoring process while still conforming to the realistic assessment range. We then aggregate 10 separate scores and compute their average to arrive at the final G-Eval score for each question-answer pair. This scoring method helps smooth out anomalies and provides a more reliable assessment of the generative model's performance, hence contributing valuable insights into its ability to mimic human-like answering proficiency.

E.2 Hyperparameters

Hyperparameter	**Value**
Temperature	2
Max_tokens	10
Top-p	1
n	10

Table 14: Hyperparameters of chat.completions.create settings for GPT-4

E.3 Used Prompt

You will be given one question and one answer. Your task is to rate the answer on one metric. Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.

Evaluation Criteria:
Factual Consistency (1-5): Does the answer contain untruthful or misleading facts that are not supported by the question?
Evaluation Method:
Rate the factual consistency of the answer on a scale of 1 to 5, where 1 indicates a low level and 5 indicates a high level of consistency with the question.
- 1: Poor. The answer contains multiple inaccuracies or misleading details.
- 3: Fair. Some elements of the answer lack factual support or contain minor inconsistencies, but the majority aligns with the question's context.
- 5: Good. The answer is factually consistent, presenting information supported by the question without any misleading details.

Question:
`{{Question}}`
Answer:
`{{Answer}}`

Evaluation Form (scores ONLY):
- Factual Consistency: